### ****Name : Abdul Rehman****

### ****Section : BSDS\_3A****

### ****Roll No : 052****

### ****Task 04 :****

### ****Part 1: Printing Only Alphabets from a String****

The first part of the program focuses on extracting only alphabetic characters from a given string that contains a mixture of letters, punctuation marks, and special symbols.

The program goes through each character in the string using a loop and checks whether it is an alphabet using the built-in isalpha() function.

* If the character is a letter (A–Z or a–z), it is printed.
* If the character is not alphabetic (for example, numbers or symbols), it is ignored.

The output displays only the alphabets, separated by spaces.  
**Purpose:**  
This section serves as a basic text filter that removes unwanted characters and keeps only alphabetic ones. Such a method is commonly used in text preprocessing, data cleaning, and basic string analysis.

### ****Part 2: Sorting Characters and Words Alphabetically****

The second part of the program deals with sorting both the individual characters and the words of a string in alphabetical order. The given string in this section is:  
"I am a student"

#### **Step 1: Sorting Characters**

1. The string is first converted into a list of characters.
2. The Bubble Sort algorithm is used to sort them alphabetically.

**Explanation of Bubble Sort:**  
Bubble Sort is a simple sorting algorithm that compares two adjacent elements at a time.

* If the first element is greater than the second, the two elements are swapped.
* This process continues until the entire list is sorted.

After sorting, the characters are joined back together into a single string.  
The output shows all the characters arranged in alphabetical order (including spaces).

#### **Step 2: Sorting Words**

1. The sentence is split into individual words using the split() function.
2. The words are then sorted alphabetically using the same Bubble Sort technique.
3. Finally, the sorted words are joined together again using spaces.

**Example:**  
Input: "I am a student"  
After sorting: "I a am student"

**Purpose:**  
This part demonstrates how sorting can be applied not only to numbers but also to text data.  
It also provides a clear example of how Bubble Sort works, making it useful for beginners learning about sorting algorithms and string manipulation.

### ****Part 3: Luhn Algorithm – Validating a Number****

The third part of the program implements the **Luhn Algorithm**, which is widely used to validate identification numbers, especially credit and debit card numbers.

#### **Steps of the Luhn Algorithm:**

1. **Reverse the number:**  
   The number is reversed to make it easier to process digits starting from the right.
2. **Double every second digit:**  
   Starting from the right (after reversing), every second digit is multiplied by 2.
3. **Adjust digits greater than 9:**  
   If doubling a digit results in a number greater than 9, 9 is subtracted from it.  
   This step ensures that the resulting digits remain single-digit numbers.
4. **Calculate the total sum:**  
   All digits (after processing) are added together to form a total.
5. **Check divisibility by 10:**  
   If the total is divisible by 10 (i.e., total % 10 == 0), the number is considered **valid**.  
   Otherwise, it is **invalid**.

#### **Example:**

For the number "5893804115457289", the algorithm performs the above steps and outputs:  
5893804115457289 is a valid number

If a number does not pass the Luhn check, the output will state that it is not valid.

**Purpose:**  
This part introduces a real-world algorithm used in financial and digital systems for verification.  
It helps students understand algorithmic logic, conditional statements, and data validation techniques.